Summation and Transduction
in Automatic Sequences

Luke Schaeffer

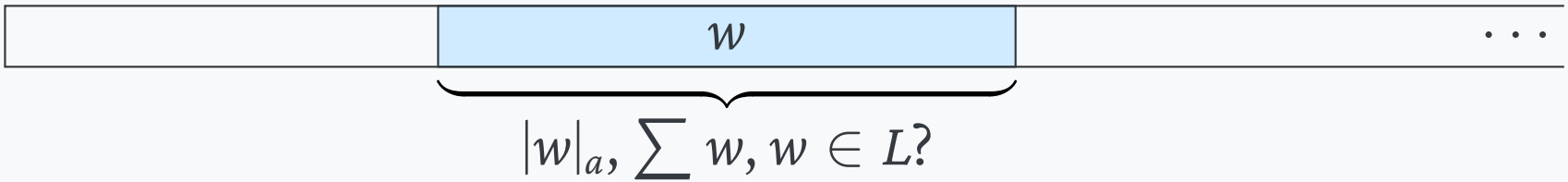
University of Waterloo

Motivation

First-order statements about k-automatic sequences can be mechanically decided by `Walnut`.

Motivation

First-order statements about k-automatic sequences can be mechanically decided by `Walnut`.



Summation/transduction are not first-order predicates.

Section 2

Summation

Period-doubling sequence

Definition

Define the period-doubling sequence,

$$\mathbf{p} \triangleq h^{\omega}(\mathtt{1}) = \mathtt{10111010101110 11} \cdots,$$

as the fixed point of $h\colon \{\mathtt{0},\mathtt{1}\}^* \to \{\mathtt{0},\mathtt{1}\}^*$ where

$$\begin{aligned} h(\mathtt{1}) &= \mathtt{10}, \\ h(\mathtt{0}) &= \mathtt{11}. \end{aligned}$$

Partial Sums

$$\mathbf{p} = \underbrace{1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1}\ 0\ 1\ 1\ \cdots$$

$$\sum$$

$$+\backslash\mathbf{p} = 0\ 1\ 1\ 2\ 3\ 4\ 4\ 5\ 5\ 6\ 6\ 7\ \cdots$$

Partial Sums

$$\mathbf{p} = \underbrace{1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1}_{\sum}\ 0\ 1\ 1\ \cdots$$

$$+\backslash\mathbf{p} = 0\ 1\ 1\ 2\ 3\ 4\ 4\ 5\ 5\ 6\ 6\ 7\ \cdots$$

Not bounded $\implies$ not automatic or morphic

Partial Sums

$$\mathbf{p} = \underbrace{1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1}_{\sum}\ 0\ 1\ 1\ \cdots$$

$$+\backslash\mathbf{p} = 0\ 1\ 1\ 2\ 3\ 4\ 4\ 5\ 5\ 6\ 6\ 7\ \cdots$$

Not bounded $\implies$ not automatic or morphic

k-automatic $\subsetneq$ k-synchronized $\subsetneq$ k-regular

Partial Sums

$$\mathbf{p} = \underbrace{1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1}_{\sum}\ 0\ 1\ 1\ \cdots$$

$$+\backslash\mathbf{p} = 0\ 1\ 1\ 2\ 3\ 4\ 4\ 5\ 5\ 6\ 6\ 7\ \cdots$$

Not bounded $\implies$ not automatic or morphic

k-automatic $\subsetneq$ k-synchronized $\subsetneq$ k-regular

Theorem

Every k-automatic sequence has k-regular partial sums.

k-regular sequences

Definition

A sequence $(a_i)_{i=0}^{\infty} \in \mathbb{Z}^{\omega}$ is k-regular if there exist

- matrices $M_0, \ldots, M_{k-1} \in \mathbb{Z}^{d \times d}$,
- vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^d$,

such that for all $n \geqslant 0$,

$$a_n = \mathbf{u}^{\top} M_{r_j} \cdots M_{r_0} \mathbf{v}.$$

where $r_j \cdots r_0 = (n)_k \in [k]^*$.

Constructing Prefixes

$$\mathbf{p}_0 \cdots \mathbf{p}_{22} \quad = \quad 10111010101110111011101$$

Constructing Prefixes

$$
\begin{aligned}
\mathbf{p}_0 \cdots \mathbf{p}_{22} &= 10111010101110111011101 \\
&= h(10111010101)1
\end{aligned}
$$

Constructing Prefixes

$$
\begin{aligned}
\mathbf{p}_0 \cdots \mathbf{p}_{22} &= 10111010101110111011101 \\
&= h(10111010101)1 \\
&= h(h(10111)1)1
\end{aligned}
$$

Constructing Prefixes

$$
\begin{aligned}
\mathbf{p}_0 \cdots \mathbf{p}_{22} &= 10111010101110111011101 \\
&= h(10111010101)1 \\
&= h(h(10111)1)1 \\
&= h(h(h(10)1)1)1
\end{aligned}
$$

Constructing Prefixes

$$
\begin{aligned}
\mathbf{p}_0 \cdots \mathbf{p}_{22} &= 10111010101110111011101 \\
&= h(10111010101)1 \\
&= h(h(10111)1)1 \\
&= h(h(h(10)1)1)1 \\
&= h(h(h(h(1))1)1)1
\end{aligned}
$$

Constructing Prefixes

$$
\begin{aligned}
\mathbf{p}_0 \cdots \mathbf{p}_{22} &= \texttt{10111010101110111011101} \\
&= h(\texttt{10111010101})\texttt{1} \\
&= h(h(\texttt{10111})\texttt{1})\texttt{1} \\
&= h(h(h(\texttt{10})\texttt{1})\texttt{1})\texttt{1} \\
&= h(h(h(h(\texttt{1}))\texttt{1})\texttt{1})\texttt{1}
\end{aligned}
$$

✗ It is not obvious which symbol(s) to append.

Constructing Prefixes

$$
\begin{aligned}
\mathbf{p}_0 \cdots \mathbf{p}_{22} &= \texttt{10111010101110111011101} \\
&= h(\texttt{10111010101})\texttt{1} \\
&= h(h(\texttt{10111})\texttt{1})\texttt{1} \\
&= h(h(h(\texttt{10})\texttt{1})\texttt{1})\texttt{1} \\
&= h(h(h(h(\texttt{1}))\texttt{1})\texttt{1})\texttt{1}
\end{aligned}
$$

✗ It is not obvious which symbol(s) to append.

✓ # of symbols comes from base-k digits

Constructing Prefixes

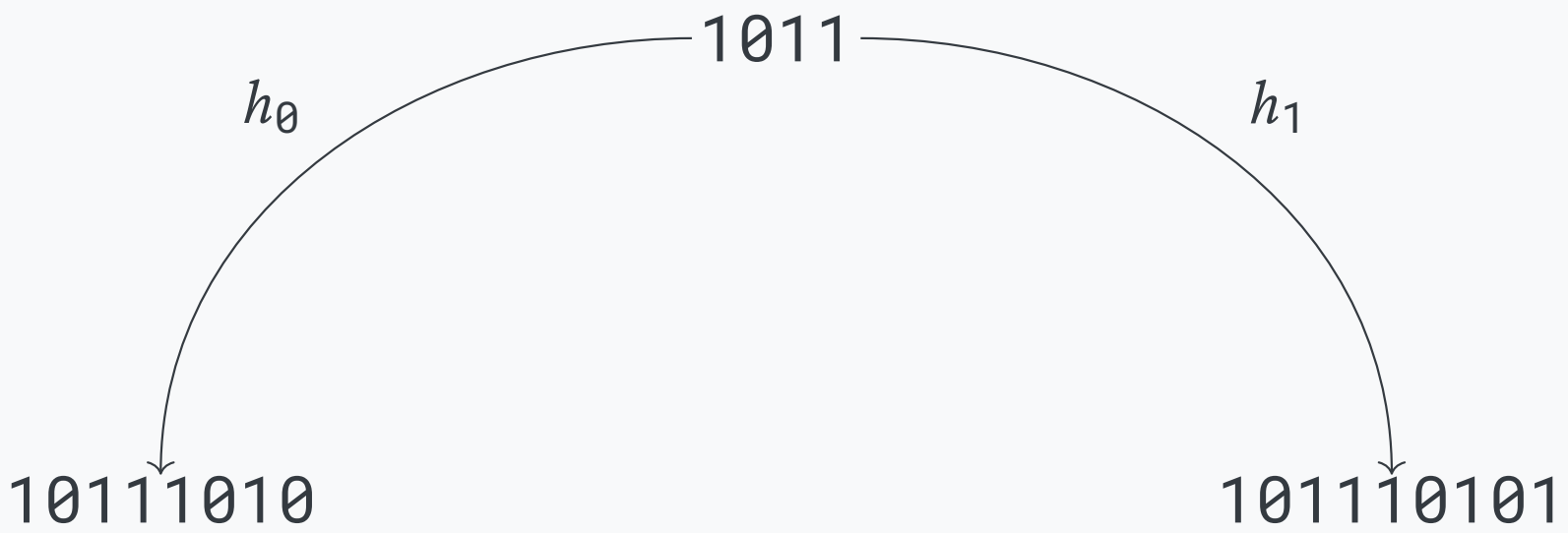
$$
\begin{aligned}
\mathbf{p}_0 \cdots \mathbf{p}_{22} &= \mathtt{10111010101110111011101} \\
&= h(\mathtt{10111010101})\mathtt{1} \\
&= h(h(\mathtt{10111})\mathtt{1})\mathtt{1} \\
&= h(h(h(\mathtt{10})\mathtt{1})\mathtt{1})\mathtt{1} \\
&= h(h(h(h(\mathtt{1}))\mathtt{1})\mathtt{1})\mathtt{1}
\end{aligned}
$$

✗ It is not obvious which symbol(s) to append.

✓ # of symbols comes from base-k digits

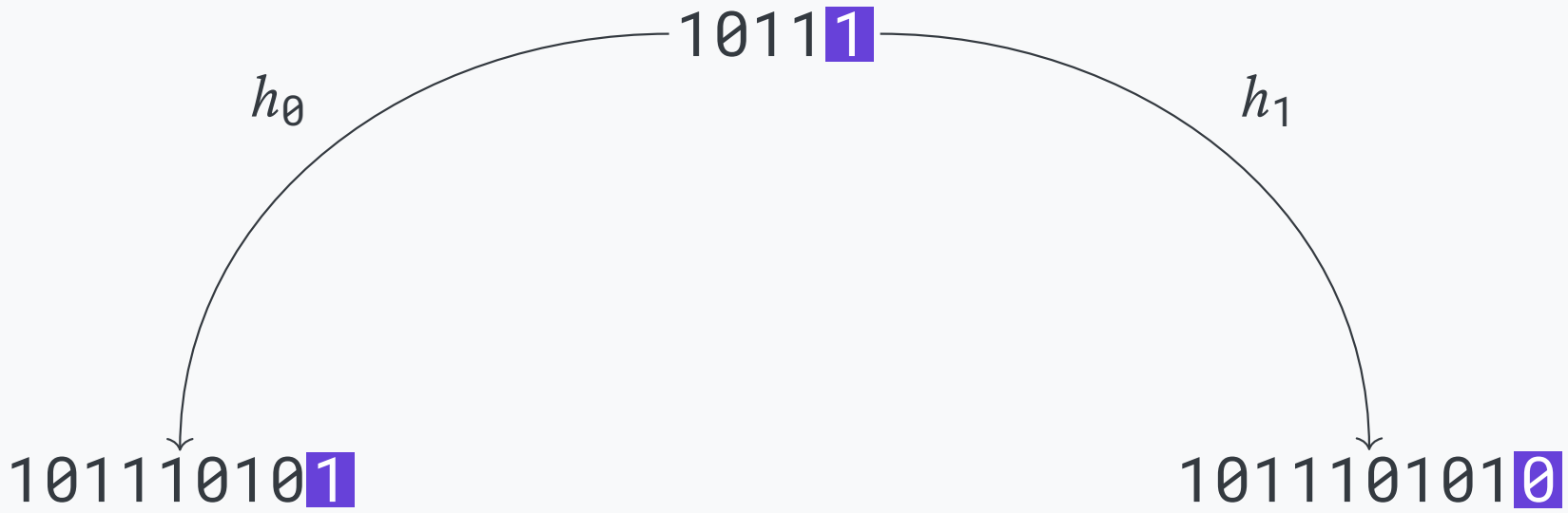
✗ We prefer not to append at all $\Longrightarrow$ pure morphic solution

Prefixes



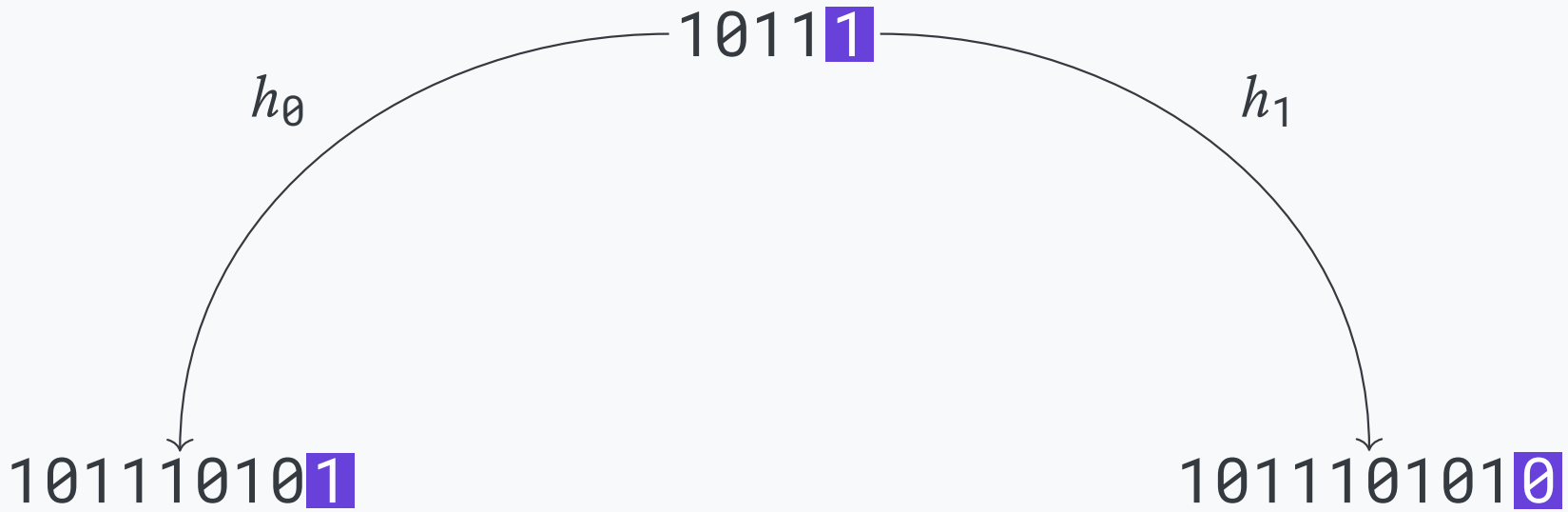
Prefixes

Include preview of next symbol



Prefixes

Include preview of next symbol



$$h_0(0) = 11 \qquad h_1(0) = 11$$
$$h_0(1) = 10 \qquad h_1(1) = 10$$
$$h_0(0) = 1 \qquad h_1(0) = 11$$
$$h_0(1) = 1 \qquad h_1(1) = 10$$

Example

Note: $(23)_2 = 10111$.

$$h_1(h_1(h_1(h_0(h_1(1)))))$$

Example

Note: $(23)_2 = 10111$.

$$h_1(h_1(h_1(h_0(h_1(1)))))$$

Example

Note: $(23)_2 = \texttt{10111}$.

$$
\begin{aligned}
& h_1(h_1(h_1(h_0(h_1(\texttt{1}))))) \\
= \; & h_1(h_1(h_1(h_0(\texttt{10})))) \\
\end{aligned}
$$

Example

Note: $(23)_2 = \texttt{10111}$.

$$
\begin{aligned}
& h_1(h_1(h_1(h_0(h_1(\texttt{1})))))\\
= \;& h_1(h_1(h_1(h_0(\texttt{10}))))\\
= \;& h_1(h_1(h_1(\texttt{101})))
\end{aligned}
$$

Example

Note: $(23)_2 = 10111$.

$$
\begin{aligned}
& h_1(h_1(h_1(h_0(h_1(1))))) \\
= \; & h_1(h_1(h_1(h_0(10)))) \\
= \; & h_1(h_1(h_1(101))) \\
= \; & h_1(h_1(10111\,0))
\end{aligned}
$$

Example

Note: $(23)_2 = 10111$.

$$
\begin{aligned}
& h_1(h_1(h_1(h_0(h_1(\boxed{1})))))\\
= \; & h_1(h_1(h_1(h_0(1\boxed{0}))))\\
= \; & h_1(h_1(h_1(10\boxed{1})))\\
= \; & h_1(h_1(10111\boxed{0}))\\
= \; & h_1(101110101\boxed{1})
\end{aligned}
$$

Example

Note: $(23)_2 = 10111$.

$$
\begin{aligned}
& h_1(h_1(h_1(h_0(h_1(\boxed{1})))))\\
= \;& h_1(h_1(h_1(h_0(1\boxed{0}))))\\
= \;& h_1(h_1(h_1(10\boxed{1})))\\
= \;& h_1(h_1(10111\boxed{0}))\\
= \;& h_1(101110101011\boxed{1})\\
= \;& 10111010101110111011101\boxed{0}
\end{aligned}
$$

Parikh vectors

Definition

The Parikh vector of a word $w \in \Sigma^*$ is

$$(|w|_a)_{a \in \Sigma}.$$

I.e., a vector of the number of occurrences of each alphabet symbol.

E.g.,

$$\psi_{\{\mathsf{A},\mathsf{B},\mathsf{N}\}}(\mathsf{BANANA}) = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$$

Why the Parikh vectors?

1. Sum reduces to Parikh vectors

$$\sum_i w_i = \mathtt{0} \cdot |w|_{\mathtt{0}} + \mathtt{1} \cdot |w|_{\mathtt{1}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}^\top \psi(w).$$

Why the Parikh vectors?

1. Sum reduces to Parikh vectors

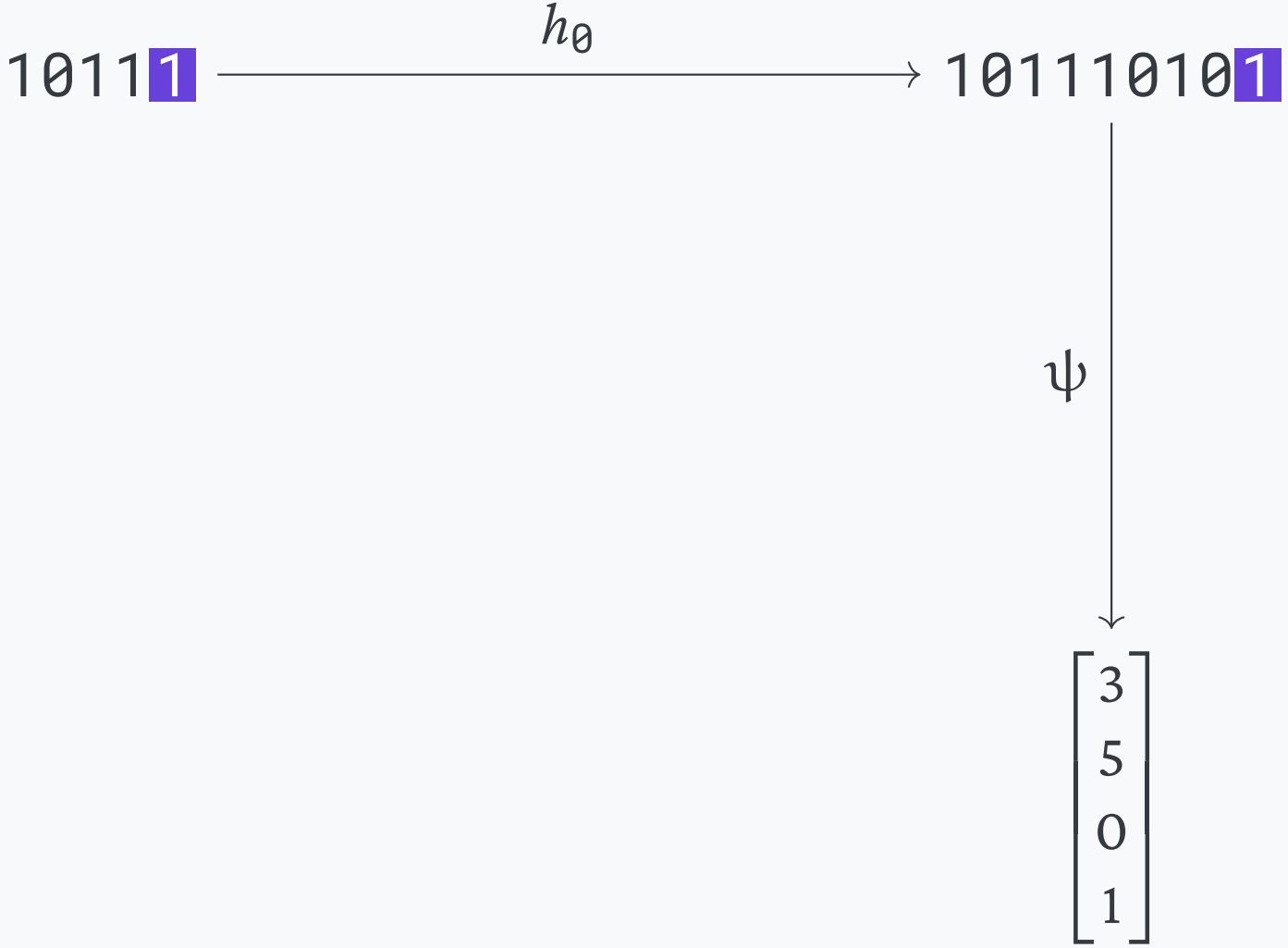
$$\sum_i w_i = \mathtt{0} \cdot |w|_{\mathtt{0}} + \mathtt{1} \cdot |w|_{\mathtt{1}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}^\top \psi(w).$$

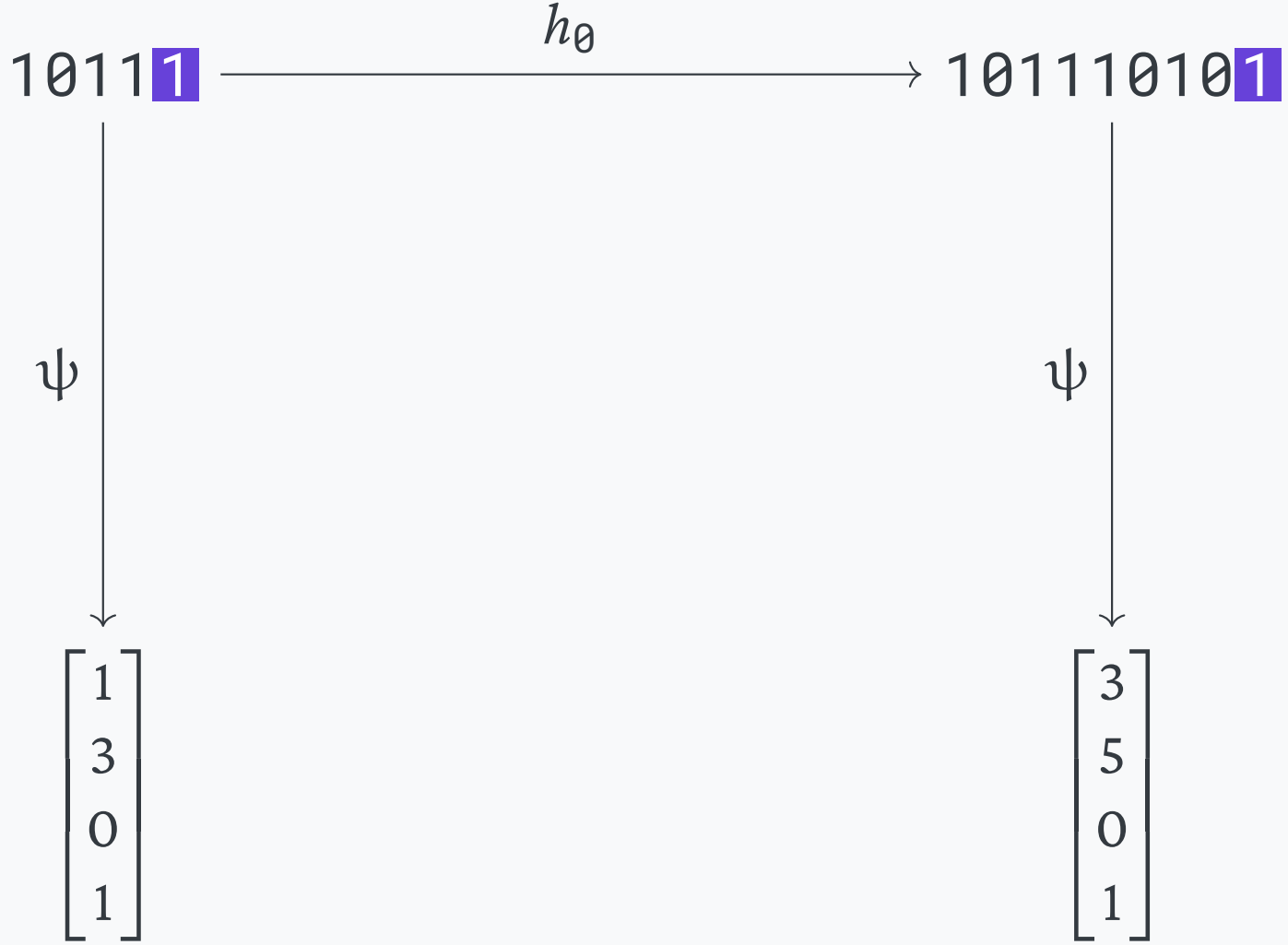
2. Diagram:

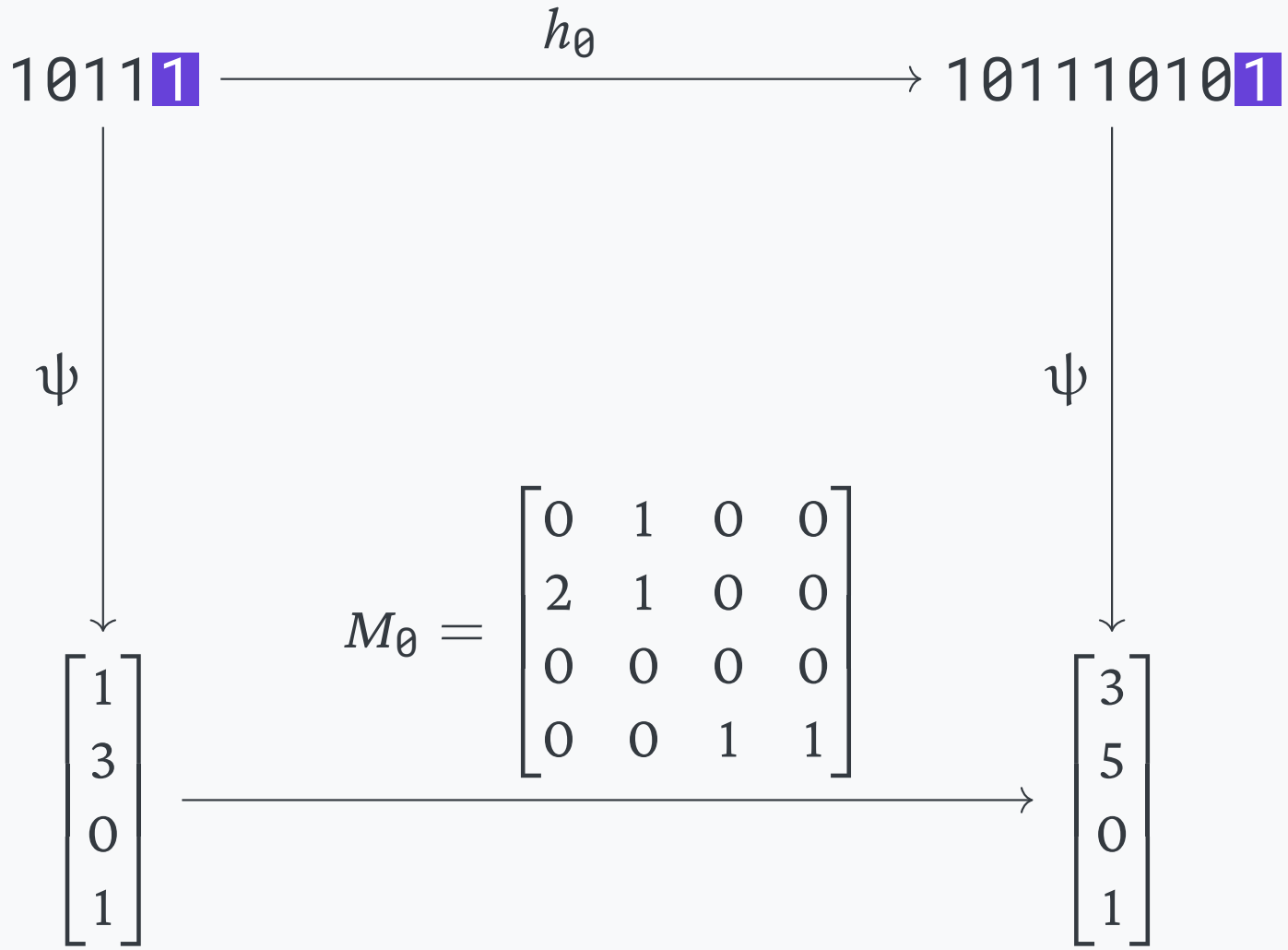
$$\begin{array}{ccc} \Sigma^* & \xrightarrow{h_i} & \Sigma^* \\ \psi \downarrow & & \downarrow \psi \\ \mathbb{N}^\Sigma & \xrightarrow[M_i]{} & \mathbb{N}^\Sigma \end{array}$$

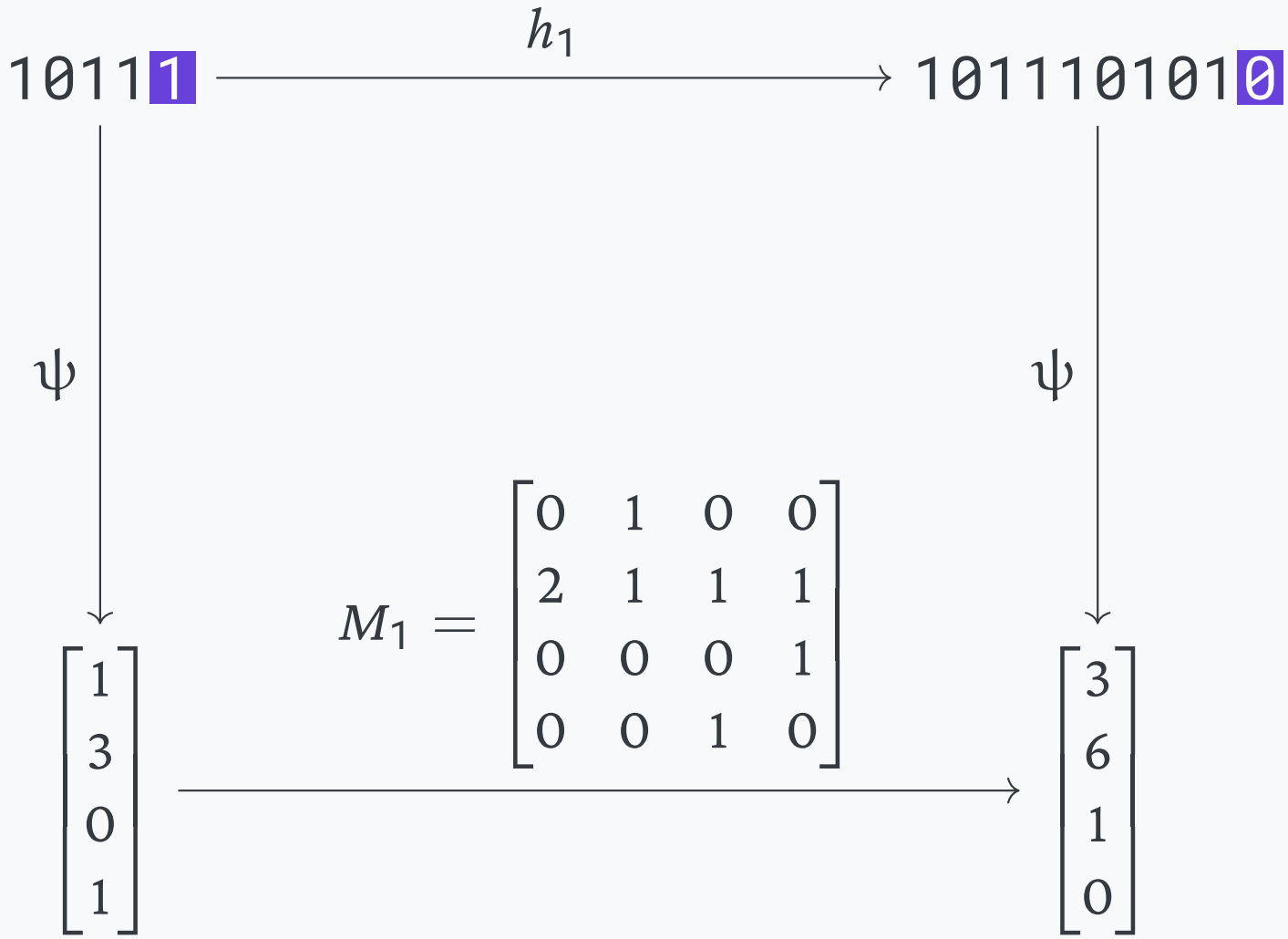
10111


h_0
10111
101110101


1011 1
h_0
101110101
ψ
3
5
0
1


h_0
10111
1
101110101
ψ
ψ
$\begin{bmatrix} 1 \\ 3 \\ 0 \\ 1 \end{bmatrix}$
$\begin{bmatrix} 3 \\ 5 \\ 0 \\ 1 \end{bmatrix}$


h_0
10111
1
101110101
ψ
ψ
$M_0 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
$\begin{bmatrix} 1 \\ 3 \\ 0 \\ 1 \end{bmatrix}$
$\begin{bmatrix} 3 \\ 5 \\ 0 \\ 1 \end{bmatrix}$


1011 1
h_1
1011101010
ψ
ψ
$M_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 2 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
$\begin{bmatrix} 1 \\ 3 \\ 0 \\ 1 \end{bmatrix}$
$\begin{bmatrix} 3 \\ 6 \\ 1 \\ 0 \end{bmatrix}$

$$\sum_{i=0}^{22} \mathbf{p}_i \quad = \quad \mathbf{u}^\top \psi(\mathbf{p}_0 \cdots \mathbf{p}_{22})$$

$$
\begin{aligned}
\sum_{i=0}^{22} \mathbf{p}_i &= \mathbf{u}^\top \psi(\mathbf{p}_0 \cdots \mathbf{p}_{22}) \\
&= \mathbf{u}^\top \psi(h_1(h_1(h_1(h_0(h_1(1)))))))
\end{aligned}
$$

$$
\begin{aligned}
\sum_{i=0}^{22} \mathbf{p}_i &= \mathbf{u}^\top \psi(\mathbf{p}_0 \cdots \mathbf{p}_{22}) \\
&= \mathbf{u}^\top \psi(h_1(h_1(h_1(h_0(h_1(\boxed{1})))))) \\
&= \mathbf{u}^\top M_1 \psi(h_1(h_1(h_0(h_1(\boxed{1})))))
\end{aligned}
$$

$$
\begin{aligned}
\sum_{i=0}^{22} \mathbf{p}_i &= \mathbf{u}^\top \psi(\mathbf{p}_0 \cdots \mathbf{p}_{22}) \\
&= \mathbf{u}^\top \psi(h_1(h_1(h_1(h_0(h_1(\boxed{1})))))) \\
&= \mathbf{u}^\top M_1 \psi(h_1(h_1(h_0(h_1(\boxed{1}))))) \\
&\quad \vdots \\
&= \mathbf{u}^\top M_1 M_1 M_1 M_0 M_1 \psi(\boxed{1})
\end{aligned}
$$

$$
\begin{aligned}
\sum_{i=0}^{22} \mathbf{p}_i &= \mathbf{u}^\top \psi(\mathbf{p}_0 \cdots \mathbf{p}_{22}) \\
&= \mathbf{u}^\top \psi(h_1(h_1(h_1(h_0(h_1(\boxed{1})))))) \\
&= \mathbf{u}^\top M_1 \psi(h_1(h_1(h_0(h_1(\boxed{1}))))) \\
&\quad \vdots \\
&= \mathbf{u}^\top M_1 M_1 M_1 M_0 M_1 \psi(\boxed{1}) \\
&= \mathbf{u}^\top M_1 M_1 M_1 M_0 M_1 \mathbf{v}
\end{aligned}
$$

$$
\begin{aligned}
\sum_{i=0}^{22} \mathbf{p}_i &= \mathbf{u}^\top \psi(\mathbf{p}_0 \cdots \mathbf{p}_{22}) \\
&= \mathbf{u}^\top \psi(h_1(h_1(h_1(h_0(h_1(1)))))) \\
&= \mathbf{u}^\top M_1 \psi(h_1(h_1(h_0(h_1(1))))) \\
&\vdots \\
&= \mathbf{u}^\top M_1 M_1 M_1 M_0 M_1 \psi(1) \\
&= \mathbf{u}^\top M_1 M_1 M_1 M_0 M_1 \mathbf{v}
\end{aligned}
$$

□

Theorem

Every k-automatic sequence has k-regular partial sums.

Warning

We cannot use a k-regular sequence directly in `Walnut`.

Reduction modulo m

Theorem

A k-regular sequence modulo m is k-automatic.

Proof

Matrices ($\mathbb{Z}^{d}$), vectors ($\mathbb{Z}^{d}$), and integers ($\mathbb{Z}$) collapse to finite sets mod m. □

Example

$$\mathbf{p} = \underbrace{1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1}_{\sum \bmod 2}\ 0\ 1\ 1\ \cdots$$

$$+\backslash\mathbf{p} \bmod 2 = 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ \cdots$$

Example

$$\mathbf{p} = \underbrace{1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1}_{\sum \bmod 2}\ 0\ 1\ 1\ \cdots$$

$$+\backslash\mathbf{p} \bmod 2 = 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ \cdots$$

Fact

For all $n \in \mathbb{N}$, $\mathbf{t}_n \equiv \sum_{i=0}^{n-1} \mathbf{p}_i \pmod 2$

Example

$$\mathbf{p} = \underbrace{1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1}_{\sum \bmod 2}\ 0\ 1\ 1\ \cdots$$

$$+\backslash\mathbf{p} \bmod 2 = 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ \cdots$$

Fact

For all $n \in \mathbb{N}$, $\mathbf{t}_n \equiv \sum_{i=0}^{n-1} \mathbf{p}_i \pmod 2$

Fact

For all $n \in \mathbb{N}$, $\mathbf{p}_n \equiv \mathbf{t}_{n+1} - \mathbf{t}_n \pmod 2$.

Finiteness

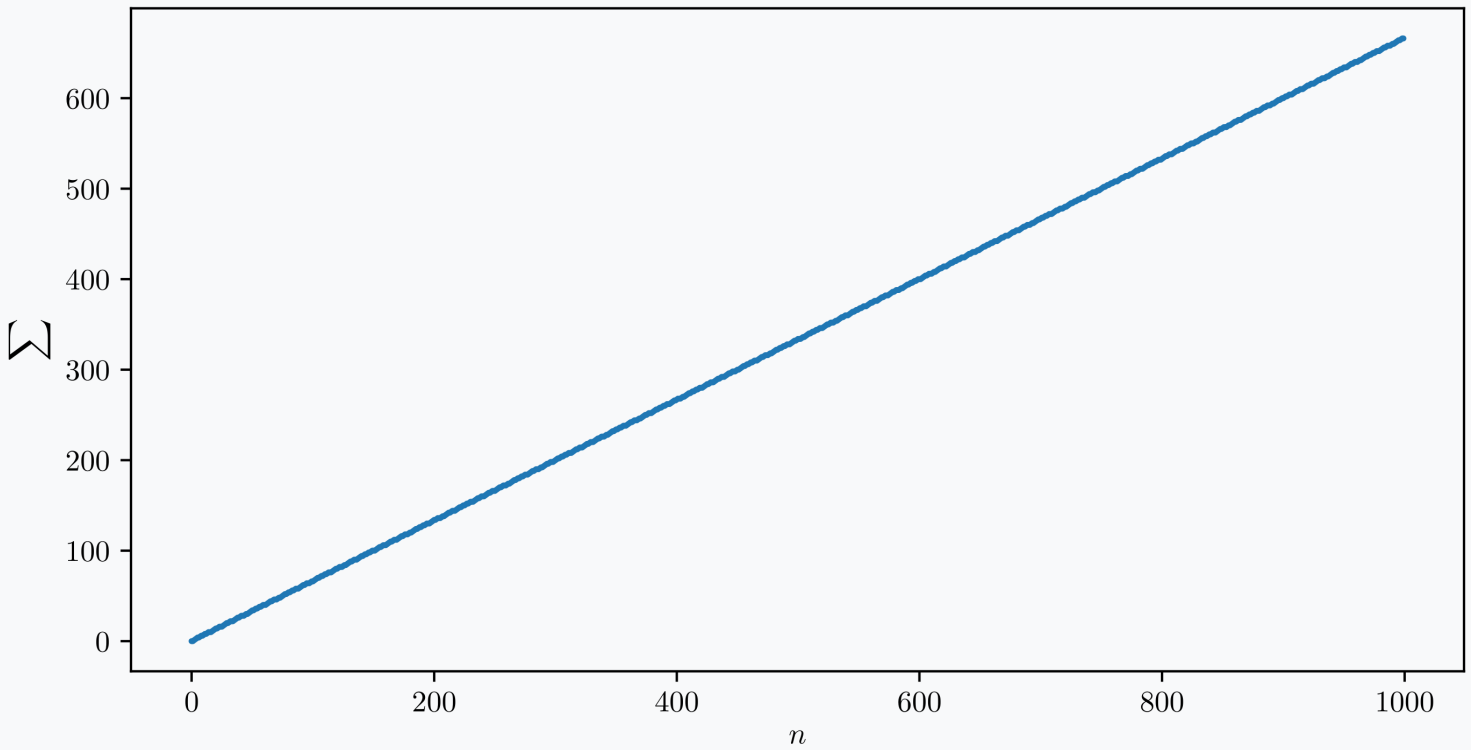
Theorem (Allouche and Shallit, 1990)

If a sequence is k-regular and takes on finitely many values then it is k-automatic.

Example

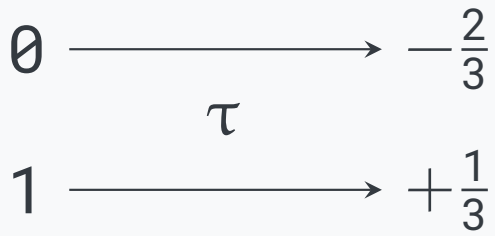
Consider $\mathbf{d} = \phi^{\omega}(1)$ where

$$\phi(0) = 011,$$
$$\phi(1) = 110.$$

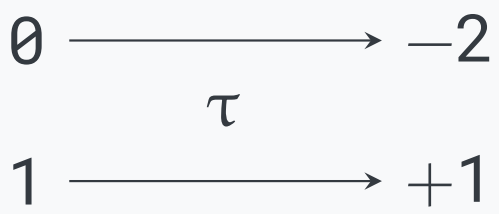


Subtract $\frac{2}{3}$ from each symbol to reduce by $\frac{2}{3}n$.

Subtract $\frac{2}{3}$ from each symbol to reduce by $\frac{2}{3}n$.

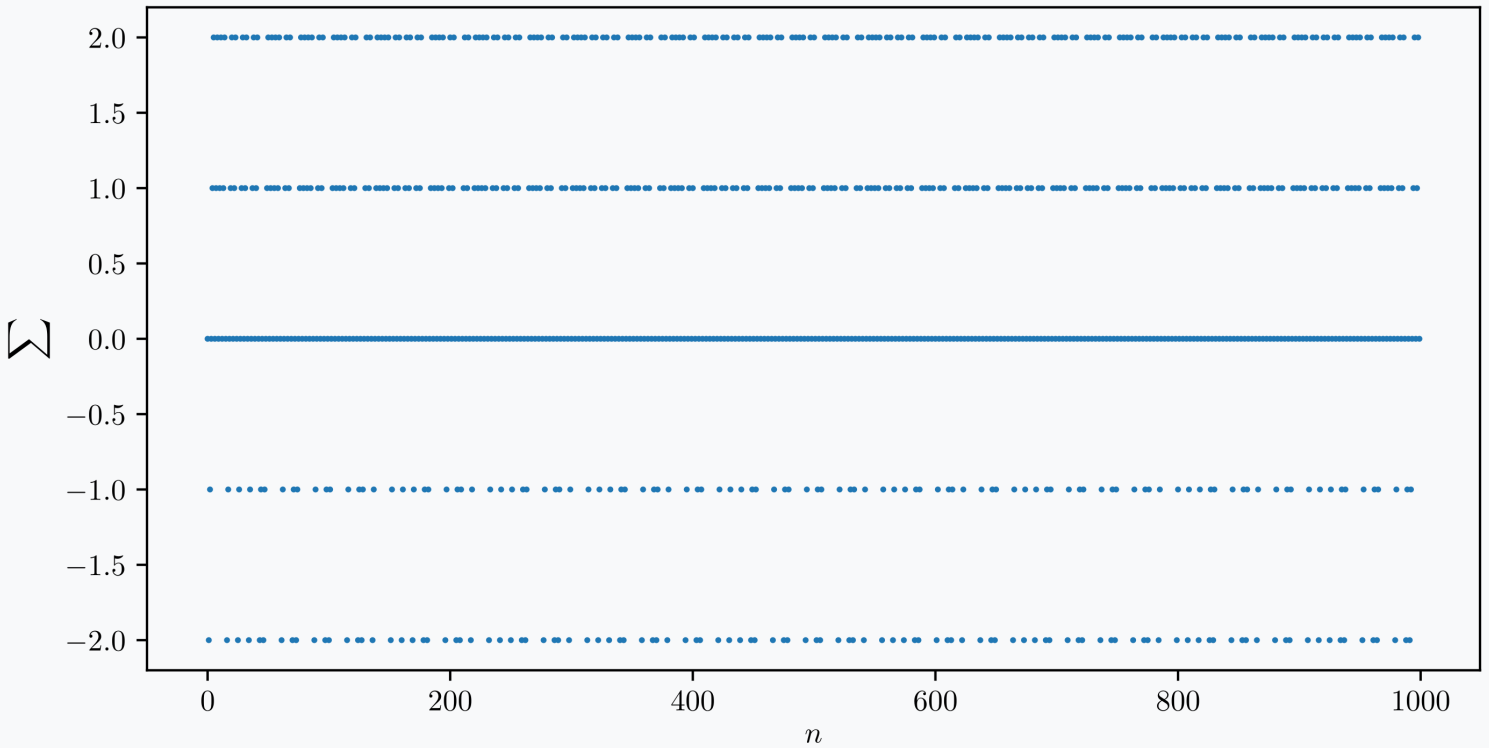


Subtract $\frac{2}{3}$ from each symbol to reduce by $\frac{2}{3}n$.



Subtract $\frac{2}{3}$ from each symbol to reduce by $\frac{2}{3}n$.

$$\begin{array}{ccc} \texttt{0} & \longrightarrow & -2 \\ & \tau & \\ \texttt{1} & \longrightarrow & +1 \end{array}$$



We have a new bounded sequence.

$$\mathbf{r}_n = \sum_{i=0}^{n-1} \tau(\mathbf{d}_i) = 3 \sum_{i=0}^{n-1} \mathbf{d}_i - 2n.$$

Theorem

The sequence $\mathbf{r}$ *is* 3*-automatic.*

k-Synchronized sequences

Definition

A function $f\colon \mathbb{N} \to \mathbb{N}$ is k-synchronized if some automaton accepts

$$\{(x, f(x))_k : x \in \mathbb{N}\}.$$

k-Synchronized sequences

Definition

A function $f\colon \mathbb{N} \to \mathbb{N}$ is k-synchronized if some automaton accepts

$$\{(x, f(x))_k : x \in \mathbb{N}\}.$$

Theorem

$n \mapsto \sum_{i=0}^{n-1} \mathbf{d}_i$ is 3-synchronized.

k-Synchronized sequences

Definition

A function $f\colon \mathbb{N} \to \mathbb{N}$ is k-synchronized if some automaton accepts

$$\{(x, f(x))_k : x \in \mathbb{N}\}.$$

Theorem

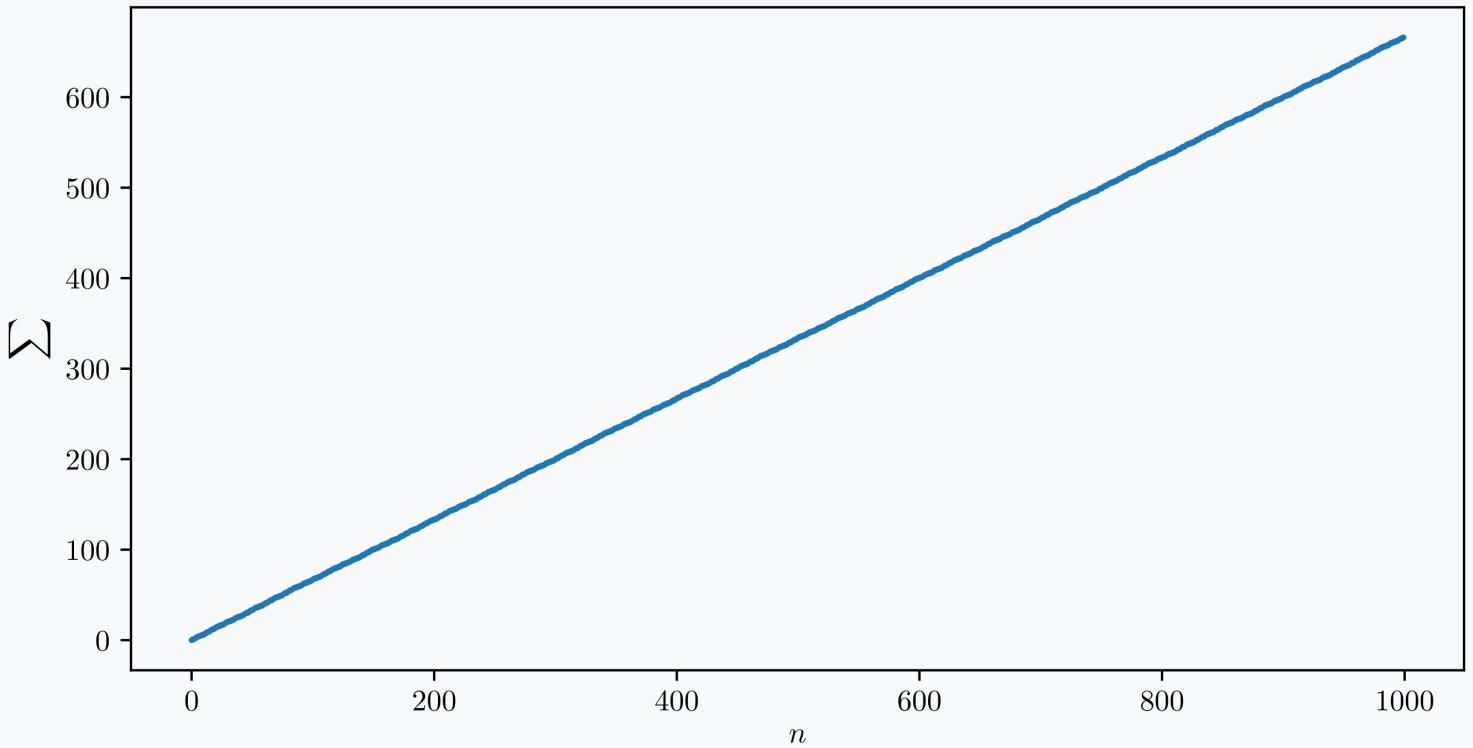
$n \mapsto \sum_{i=0}^{n-1} \mathbf{d}_i$ *is* 3*-synchronized.*

Given (n, s), check if

$$\mathbf{r}_n = 3s - 2n.$$

Non-example

Partial sums of $\mathbf{p}$ also tend to line of slope $\frac{2}{3}$.

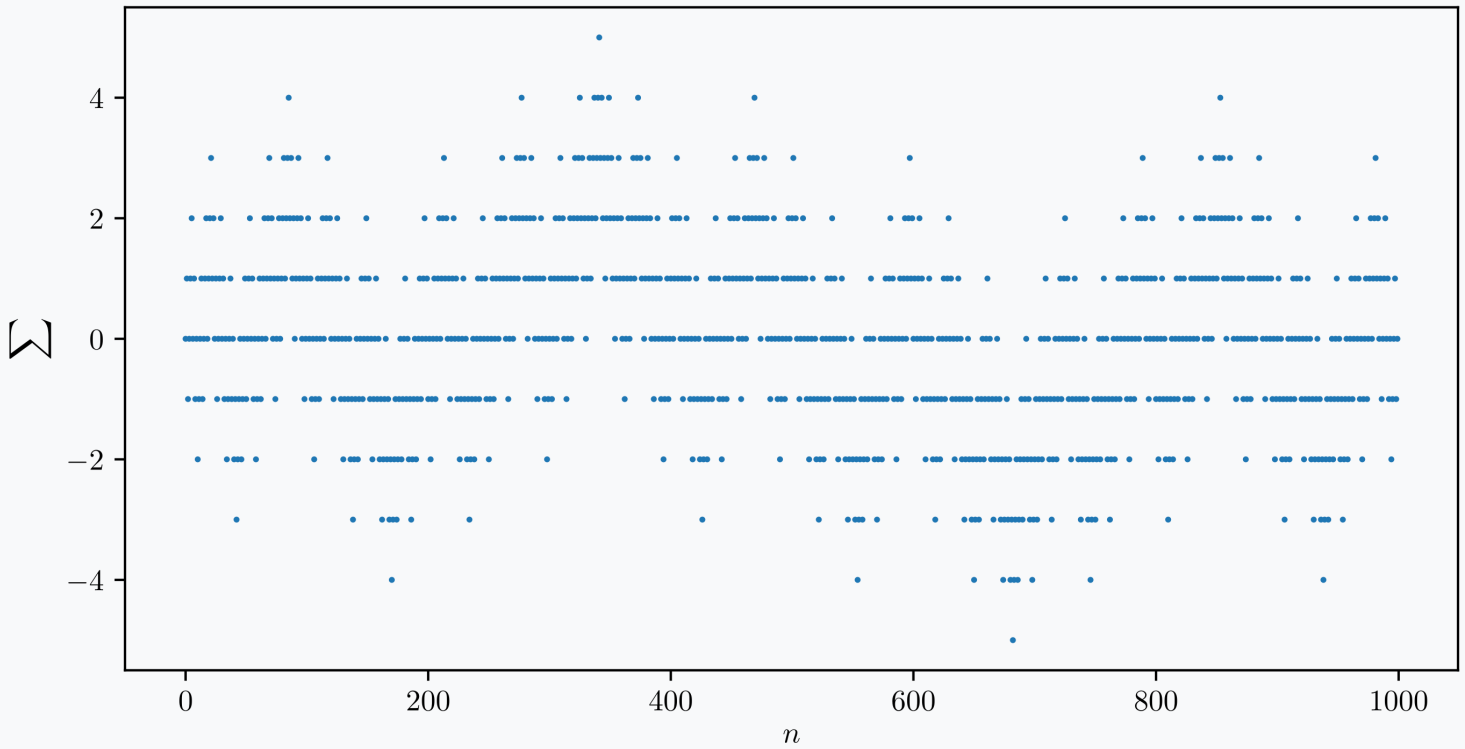


Non-example

Apply τ to subtract $\frac{2}{3}n$:

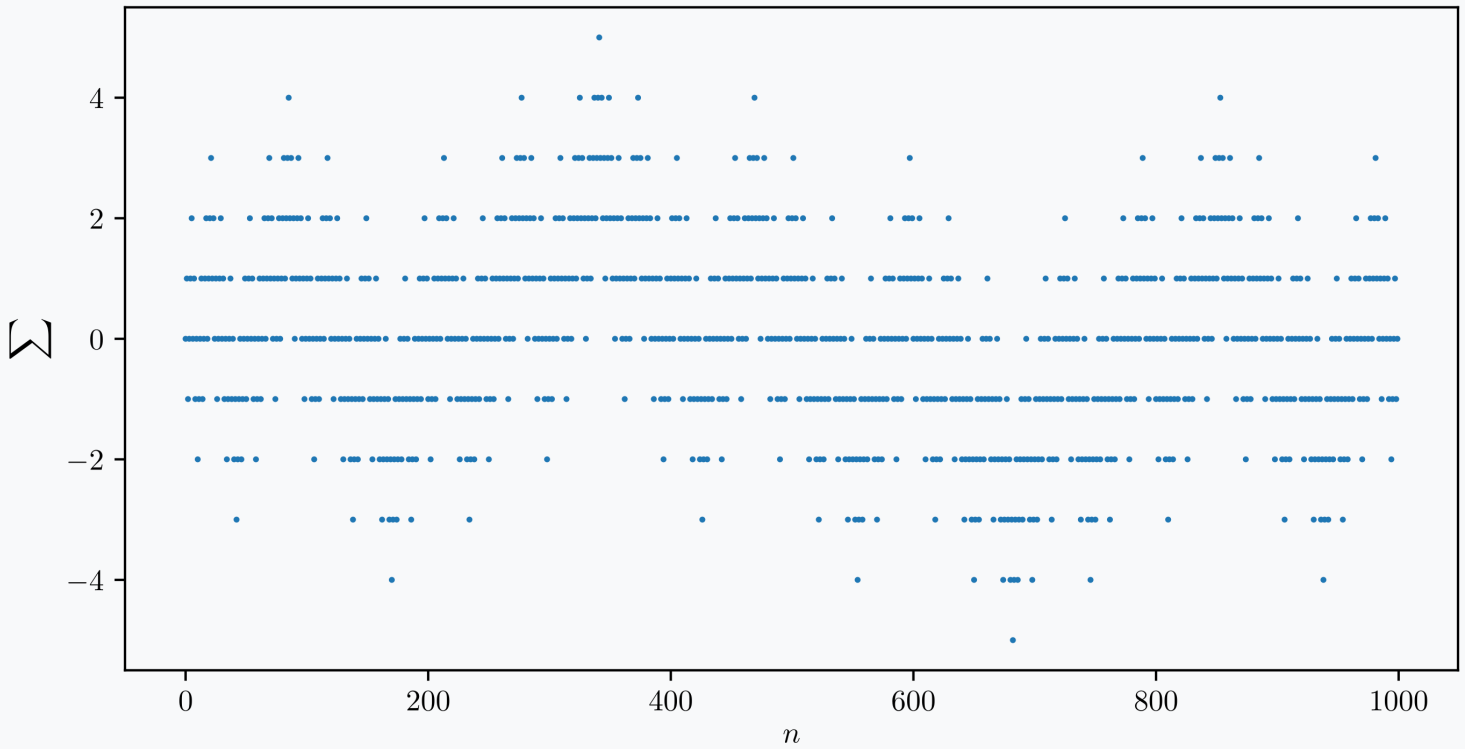
Non-example

Apply τ to subtract $\frac{2}{3}n$:



Non-example

Apply τ to subtract $\frac{2}{3}n$:



Grows like $\mathcal{O}(\log n)$.

Not synchronized either

$$\sum_{i=0}^{n-1} \mathbf{p}_i \text{ is 2-synchronized} \iff \sum_{i=0}^{n-1} \tau(\mathbf{p}_i) \text{ is 2-synchronized}$$

Not synchronized either

$$\sum_{i=0}^{n-1} \mathbf{p}_i \text{ is 2-synchronized} \iff \sum_{i=0}^{n-1} \tau(\mathbf{p}_i) \text{ is 2-synchronized}$$

Theorem

If k-synchronized sequence is $o(n)$ then it is $\mathcal{O}(1)$.

Beatty sequences

Theorem (S., Shallit, Zorcic 2024)

Let $0 < \gamma < 1$ be a quadratic irrational, and suppose $\alpha, \beta \in \mathbb{Q}(\gamma)$ are such that $\alpha \geqslant 0$, $\alpha + \beta \geqslant 0$. Then

$$(\lfloor \alpha n + \beta \rfloor)_{n \geqslant 1}^{\infty}$$

is synchronized in γ-Ostrowski representation.

Beatty sequences

Theorem (S., Shallit, Zorcic 2024)

Let $0 < \gamma < 1$ be a quadratic irrational, and suppose $\alpha, \beta \in \mathbb{Q}(\gamma)$ are such that $\alpha \geqslant 0$, $\alpha + \beta \geqslant 0$. Then

$$(\lfloor \alpha n + \beta \rfloor)_{n \geqslant 1}^{\infty}$$

is synchronized in γ-Ostrowski representation.

In other words, partial sums of Sturmian words are synchronized.

Section 3

Transduction

Theorem

Let $\mathbf{w} \in \Sigma^\omega$ *be a* k*-automatic sequence. Given any DFAO* T*, the sequence*

$$T(w_1),\ T(w_1w_2),\ T(w_1w_2w_3),\ \ldots$$

is k*-automatic.*

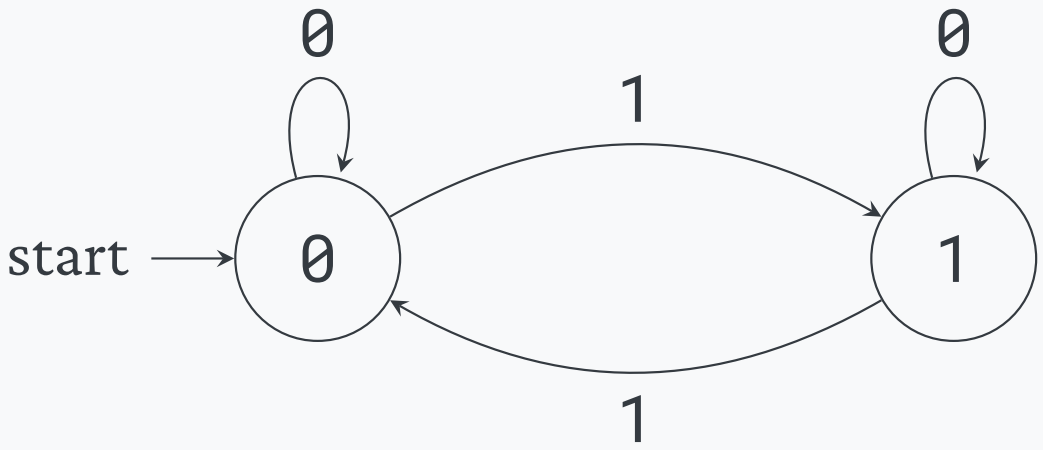
Theorem

Let $\mathbf{w} \in \Sigma^\omega$ *be a k-automatic sequence. Given any DFAO T, the sequence*

$$T(w_1), T(w_1w_2), T(w_1w_2w_3), \ldots$$

is k-automatic.

E.g., with the DFAO computing parity.



Theorem (Dekking, 1991)

Let $\mathbf{f} \in (F \to F)^\omega$ *be a* k*-automatic sequence of functions on a finite set* F*.*

$$\mathbf{f}_1, \mathbf{f}_2 \circ \mathbf{f}_1, \mathbf{f}_3 \circ \mathbf{f}_2 \circ \mathbf{f}_1, \ldots$$

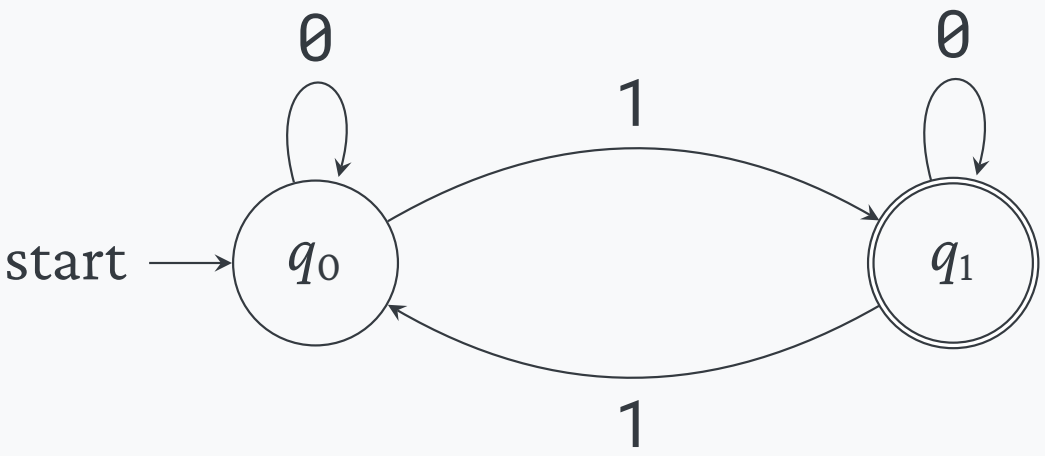
is k*-automatic.*

Theorem (Dekking, 1991)

Let $\mathbf{f} \in (F \to F)^{\omega}$ *be a k-automatic sequence of functions on a finite set F.*

$$\mathbf{f}_1, \mathbf{f}_2 \circ \mathbf{f}_1, \mathbf{f}_3 \circ \mathbf{f}_2 \circ \mathbf{f}_1, \ldots$$

is k-automatic.

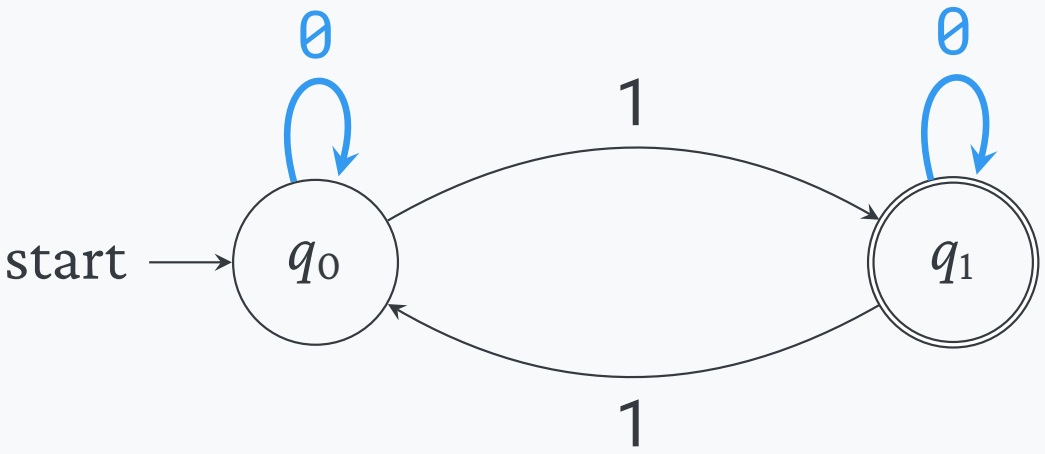


Theorem (Dekking, 1991)

Let $\mathbf{f} \in (F \to F)^{\omega}$ *be a k-automatic sequence of functions on a finite set* F*.*

$$\mathbf{f}_1, \mathbf{f}_2 \circ \mathbf{f}_1, \mathbf{f}_3 \circ \mathbf{f}_2 \circ \mathbf{f}_1, \ldots$$

is k-automatic.

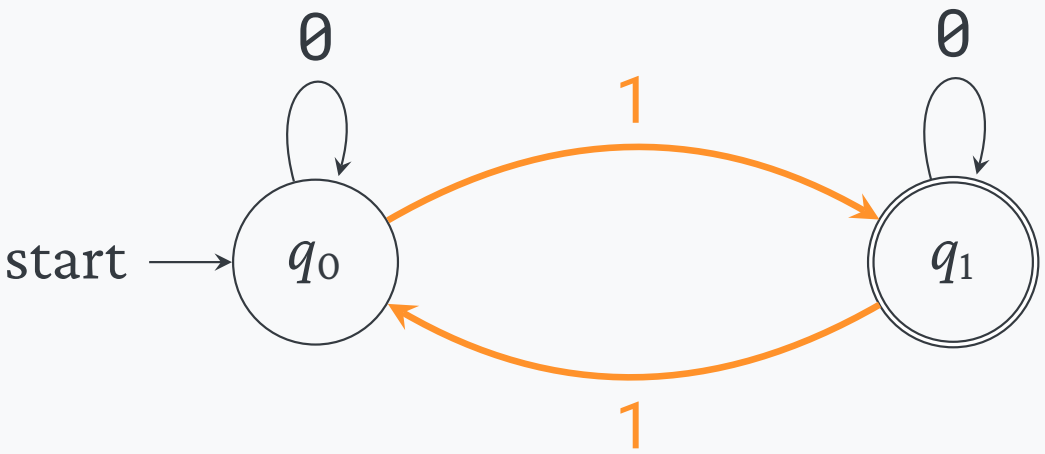


Theorem (Dekking, 1991)

Let $\mathbf{f} \in (F \to F)^{\omega}$ *be a k-automatic sequence of functions on a finite set* F*.*

$$\mathbf{f}_1, \mathbf{f}_2 \circ \mathbf{f}_1, \mathbf{f}_3 \circ \mathbf{f}_2 \circ \mathbf{f}_1, \ldots$$

is k-automatic.



Theorem

Let $\mathbf{w} \in \Sigma^\omega$ *be a k-automatic sequence. Given*

- *a finite monoid* $(M, \cdot)$*, and*
- *homomorphism* $\phi\colon \Sigma^* \to M$*,*

the sequence

$$\phi(w_1), \phi(w_1 w_2), \phi(w_1 w_2 w_3), \ldots$$

is k-automatic.

Theorem

Let $\mathbf{w} \in \Sigma^\omega$ *be a k-automatic sequence. Given*

- *a finite monoid* $(M, \cdot)$*, and*
- *homomorphism* $\phi \colon \Sigma^* \to M$*,*

the sequence

$$\phi(w_1), \phi(w_1 w_2), \phi(w_1 w_2 w_3), \ldots$$

is k-automatic.

Reduction: $(F \to F, \circ)$ is a finite monoid when F is finite.

Theorem

Let $\mathbf{w} \in \Sigma^\omega$ *be a k-automatic sequence. Given*

- *a finite monoid* $(M, \cdot)$*, and*
- *homomorphism* $\phi \colon \Sigma^* \to M$*,*

the sequence

$$\phi(w_1), \phi(w_1 w_2), \phi(w_1 w_2 w_3), \ldots$$

is k-automatic.

Reduction: $(F \to F, \circ)$ is a finite monoid when F is finite.

Reduction: Can compute ϕ with finite state: M.

Typical term

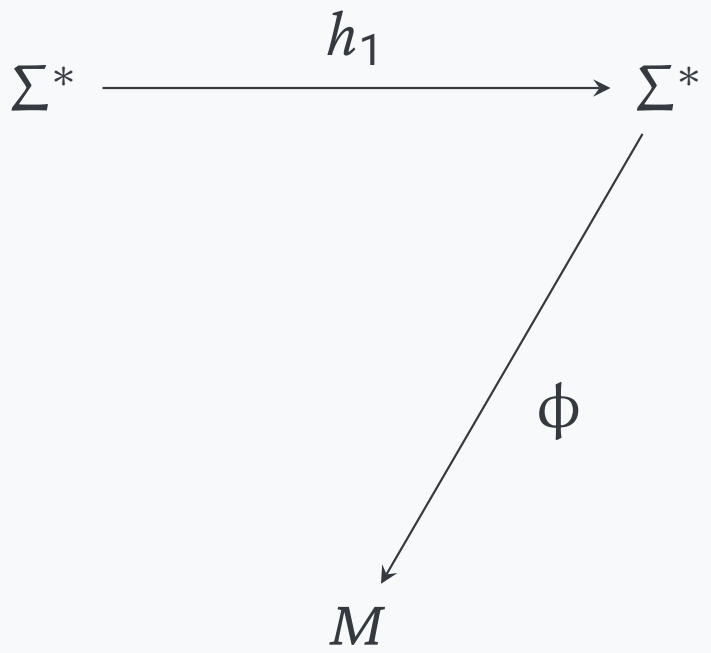
For $\mathbf{p}$, a typical term looks like

$$\phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))].$$

Typical term

For $\mathbf{p}$, a typical term looks like

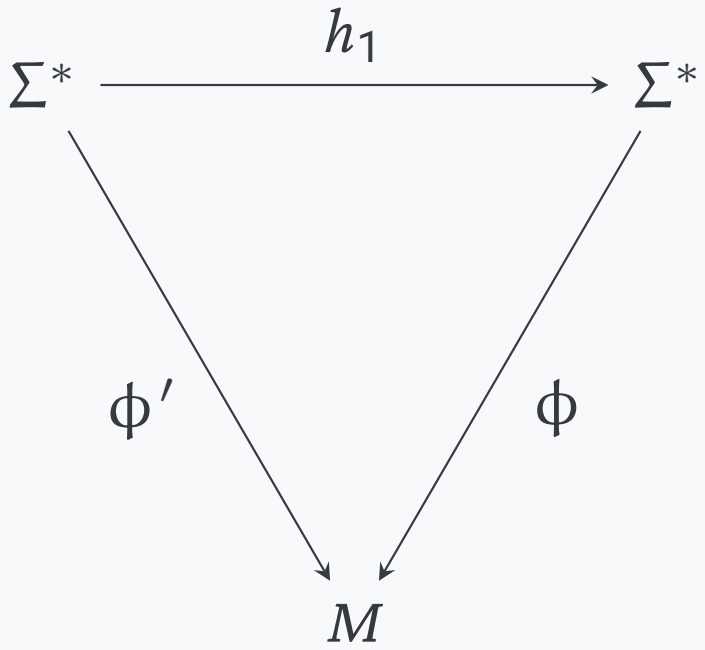
$$\phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))].$$



Typical term

For $\mathbf{p}$, a typical term looks like

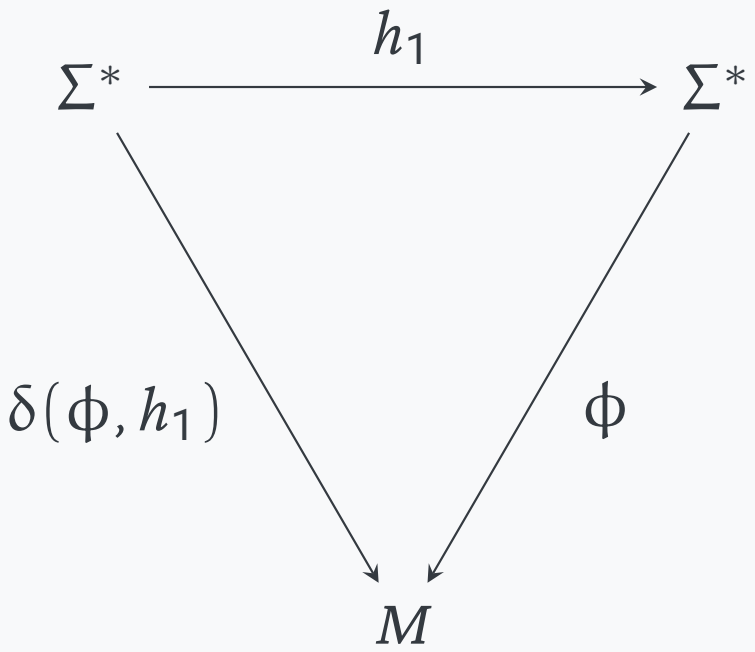
$$\phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))].$$



Typical term

For $\mathbf{p}$, a typical term looks like

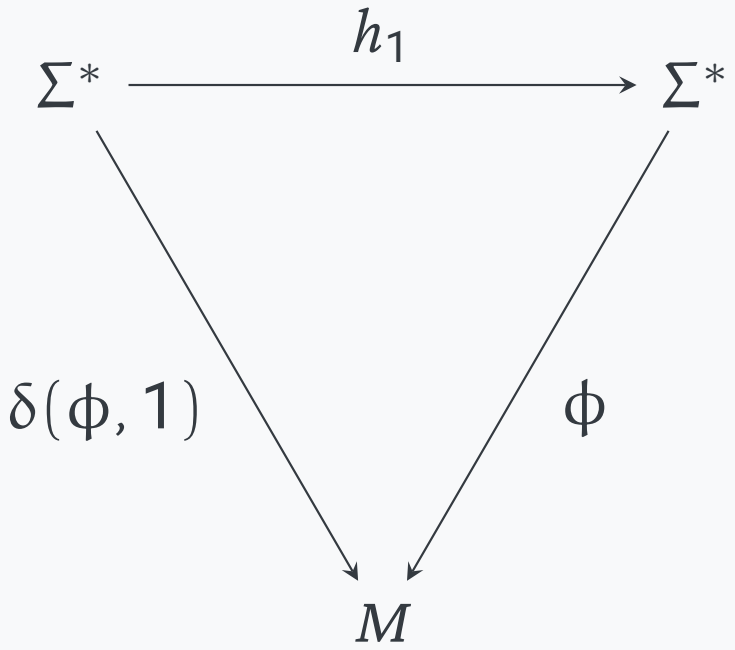
$$\phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))].$$



Typical term

For $\mathbf{p}$, a typical term looks like

$$\phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))].$$



$$\phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))]$$

$$
\begin{aligned}
& \phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))] \\
= \quad & \delta(\phi, 1)[h_1(h_1(h_0(h_1(\boxed{1}))))]
\end{aligned}
$$

$$
\begin{aligned}
& \phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))] \\
= \quad & \delta(\phi, 1)[h_1(h_1(h_0(h_1(\boxed{1}))))] \\
= \quad & \delta(\delta(\phi, 1), 1)[h_1(h_0(h_1(\boxed{1})))]
\end{aligned}
$$

$$
\begin{aligned}
& \phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))] \\
= \quad & \delta(\phi, 1)[h_1(h_1(h_0(h_1(\boxed{1}))))] \\
= \quad & \delta(\delta(\phi, 1), 1)[h_1(h_0(h_1(\boxed{1})))] \\
= \quad & \delta(\delta(\delta(\phi, 1), 1), 1)[h_0(h_1(\boxed{1}))]
\end{aligned}
$$

$$
\begin{aligned}
& \phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))] \\
= \quad & \delta(\phi, 1)[h_1(h_1(h_0(h_1(\boxed{1}))))] \\
= \quad & \delta(\delta(\phi, 1), 1)[h_1(h_0(h_1(\boxed{1})))] \\
= \quad & \delta(\delta(\delta(\phi, 1), 1), 1)[h_0(h_1(\boxed{1}))] \\
= \quad & \delta(\delta(\delta(\delta(\phi, 1), 1), 1), 0)[h_1(\boxed{1})]
\end{aligned}
$$

$$
\begin{aligned}
& \phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))] \\
= \quad & \delta(\phi, 1)[h_1(h_1(h_0(h_1(\boxed{1}))))] \\
= \quad & \delta(\delta(\phi, 1), 1)[h_1(h_0(h_1(\boxed{1})))] \\
= \quad & \delta(\delta(\delta(\phi, 1), 1), 1)[h_0(h_1(\boxed{1}))] \\
= \quad & \delta(\delta(\delta(\delta(\phi, 1), 1), 1), 0)[h_1(\boxed{1})] \\
= \quad & \delta(\delta(\delta(\delta(\delta(\phi, 1), 1), 1), 0), 1)[\boxed{1}]
\end{aligned}
$$

$$
\begin{aligned}
& \phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))] \\
= \quad & \delta(\phi, 1)[h_1(h_1(h_0(h_1(\boxed{1}))))] \\
= \quad & \delta(\delta(\phi, 1), 1)[h_1(h_0(h_1(\boxed{1})))] \\
= \quad & \delta(\delta(\delta(\phi, 1), 1), 1)[h_0(h_1(\boxed{1}))] \\
= \quad & \delta(\delta(\delta(\delta(\phi, 1), 1), 1), 0)[h_1(\boxed{1})] \\
= \quad & \delta(\delta(\delta(\delta(\delta(\phi, 1), 1), 1), 0), 1)[\boxed{1}] \\
= \quad & \delta^*(\phi, 11101)[\boxed{1}]
\end{aligned}
$$

$$
\begin{aligned}
& \phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))] \\
= \;& \delta(\phi, 1)[h_1(h_1(h_0(h_1(\boxed{1}))))] \\
= \;& \delta(\delta(\phi, 1), 1)[h_1(h_0(h_1(\boxed{1})))] \\
= \;& \delta(\delta(\delta(\phi, 1), 1), 1)[h_0(h_1(\boxed{1}))] \\
= \;& \delta(\delta(\delta(\delta(\phi, 1), 1), 1), 0)[h_1(\boxed{1})] \\
= \;& \delta(\delta(\delta(\delta(\delta(\phi, 1), 1), 1), 0), 1)[\boxed{1}] \\
= \;& \delta^*(\phi, 11101)[\boxed{1}]
\end{aligned}
$$

$$
\delta \colon \mathrm{Mon}(\Sigma^*, M) \times \{0, 1\} \to \mathrm{Mon}(\Sigma^*, M)
$$

$$
\begin{aligned}
& \phi[h_1(h_1(h_1(h_0(h_1(\boxed{1})))))] \\
= \;& \delta(\phi, 1)[h_1(h_1(h_0(h_1(\boxed{1}))))] \\
= \;& \delta(\delta(\phi, 1), 1)[h_1(h_0(h_1(\boxed{1})))] \\
= \;& \delta(\delta(\delta(\phi, 1), 1), 1)[h_0(h_1(\boxed{1}))] \\
= \;& \delta(\delta(\delta(\delta(\phi, 1), 1), 1), 0)[h_1(\boxed{1})] \\
= \;& \delta(\delta(\delta(\delta(\delta(\phi, 1), 1), 1), 0), 1)[\boxed{1}] \\
= \;& \delta^*(\phi, 11101)[\boxed{1}]
\end{aligned}
$$

$$
\delta \colon \mathrm{Mon}(\Sigma^*, M) \times \{0, 1\} \to \mathrm{Mon}(\Sigma^*, M)
$$

Note: $\mathrm{Mon}(\Sigma^*, M) \cong \mathrm{Set}(\Sigma, M)$ is finite!

Finite automaton data

- Input alphabet: $\Sigma = \{\mathtt{0}, \mathtt{1}\}$,
- Output alphabet: M,
- State set: $Q := \mathrm{Mon}(\Sigma^*, M)$,
- Transition function: $\delta \colon Q \times \{\mathtt{0}, \mathtt{1}\} \to Q$,

Finite automaton data

- Input alphabet: $\Sigma = \{\mathtt{0}, \mathtt{1}\}$,
- Output alphabet: M,
- State set: $Q := \mathrm{Mon}(\Sigma^*, M)$,
- Transition function: $\delta \colon Q \times \{\mathtt{0}, \mathtt{1}\} \to Q$,
- Initial state: $q_0 := \phi \in Q$,

Finite automaton data

- Input alphabet: $\Sigma = \{\mathtt{0}, \mathtt{1}\}$,
- Output alphabet: M,
- State set: $Q := \mathrm{Mon}(\Sigma^*, M)$,
- Transition function: $\delta \colon Q \times \{\mathtt{0}, \mathtt{1}\} \to Q$,
- Initial state: $q_0 := \phi \in Q$,
- Output map: $\tau \colon Q \to M$ where $\tau(q) = q(\mathtt{1})$.

Finite automaton data

- Input alphabet: $\Sigma = \{0, 1\}$,
- Output alphabet: M,
- State set: $Q := \mathrm{Mon}(\Sigma^*, M)$,
- Transition function: $\delta\colon Q \times \{0, 1\} \to Q$,
- Initial state: $q_0 := \phi \in Q$,
- Output map: $\tau\colon Q \to M$ where $\tau(q) = q(1)$.

□

Theorem

Let $\mathbf{w} \in \Sigma^\omega$ *be a k-automatic sequence. Given any DFAO T, the sequence*

$$T(w_1), T(w_1w_2), T(w_1w_2w_3), \ldots$$

is k-automatic.

Transducing k-automatic sequences

Walnut update March 27, 2023

One may now transduce automata (that have at most one edge per input per two states) with the following syntax:

```
transduce <new> <TRANSDUCER> <old>
```

For example, to transduce a word automaton T saved in . . . using a transducer named RUNSUM2 saved in . . ., one writes the following:

```
transduce new_T RUNSUM2 T;
```

Section 4

Something New

The morphisms h_0, h_1 are arbitrary!

The morphisms h_0, h_1 are arbitrary!

Lemma

There exists a DFAO computing

$$\phi(h_{a_1}(h_{a_2}(\cdots h_{a_k}(\sigma)\cdots)))$$

given input $a_1 \cdots a_k \in \Gamma^*$ *where*

The morphisms h_0, h_1 are arbitrary!

Lemma

There exists a DFAO computing

$$\phi(h_{a_1}(h_{a_2}(\cdots h_{a_k}(\sigma)\cdots)))$$

given input $a_1 \cdots a_k \in \Gamma^*$ where

- Σ, Γ *are finite sets,*
- $(h_\gamma : \Sigma^* \to \Sigma^*)_{\gamma\in\Gamma}$
- *a word $\sigma \in \Sigma^*$,*
- $(M, \cdot)$ *is a finite monoid, and*
- $\phi : \Sigma^* \to M$ *is a monoid homomorphism.*

Corollary (Transducing factors–Jason Bell, S., 2013)

Let $\mathbf{w} \in \Sigma^*$ *be a* k*-automatic sequence. For any DFAO* T*, the 2D sequence*

$$\mathbf{a}_{i,n} := T\big(\mathbf{w}_i \cdots \mathbf{w}_{i+n-1}\big),$$

is k*-automatic.*

Paperfolding Sequences

$\wedge\wedge\vee\wedge\wedge\vee\vee$

Definition

The regular paperfolding word,

$$\mathbf{pf} := \wedge\wedge\vee\wedge\wedge\vee\vee\wedge\wedge\wedge\vee\vee\wedge\vee\vee\cdots,$$

is defined as the unique fixed point of

$$\mathbf{pf} = \text{interleave}(\wedge\vee\wedge\vee\wedge\vee\cdots, \mathbf{pf}).$$

Definition

The regular paperfolding word,

$$\mathbf{pf} := \wedge\wedge\vee\wedge\wedge\vee\vee\wedge\wedge\wedge\vee\vee\wedge\vee\vee\cdots,$$

is defined as the unique fixed point of

$$\mathbf{pf} = \mathsf{interleave}(\wedge\vee\wedge\vee\wedge\vee\cdots, \mathbf{pf}).$$

Fact

The regular paperfolding word is 2-automatic.

Every time you unfold you make a choice, $\bigwedge$ or $\bigvee$.

Every time you unfold you make a choice, $\wedge$ or $\vee$.

The first unfold fixes the word to be of the form

$$\mathsf{interleave}(\wedge\ \vee\ \wedge\ \vee\ \wedge\ \vee\ \cdots\ , \_)$$

OR

$$\mathsf{interleave}(\vee\ \wedge\ \vee\ \wedge\ \vee\ \wedge\ \cdots\ , \_)$$

Every time you unfold you make a choice, $\wedge$ or $\vee$.

The first unfold fixes the word to be of the form

$$\text{interleave}(\wedge \vee \wedge \vee \wedge \vee \cdots, -)$$

OR

$$\text{interleave}(\vee \wedge \vee \wedge \vee \wedge \cdots, -)$$

With some work, this can be translated to morphisms.

$$h_\wedge(\mathsf{S}) = \mathsf{SA} \qquad h_\vee(\mathsf{S}) = \mathsf{SC}$$
$$h_\wedge(\mathsf{A}) = \mathsf{BC} \qquad h_\vee(\mathsf{A}) = \mathsf{BA}$$
$$h_\wedge(\mathsf{B}) = \mathsf{BA} \qquad h_\vee(\mathsf{B}) = \mathsf{BC}$$
$$h_\wedge(\mathsf{C}) = \mathsf{DC} \qquad h_\vee(\mathsf{C}) = \mathsf{DA}$$
$$h_\wedge(\mathsf{D}) = \mathsf{DA} \qquad h_\vee(\mathsf{D}) = \mathsf{DC}$$

over internal alphabet $\{\mathsf{S}, \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$ and a coding

$$\tau(\mathsf{S}) = \square$$
$$\tau(\mathsf{A}) = \wedge$$
$$\tau(\mathsf{B}) = \wedge$$
$$\tau(\mathsf{C}) = \vee$$
$$\tau(\mathsf{D}) = \vee$$

Finite paperfolding words

$$\tau(h_{\wedge}(h_{\wedge}(h_{\wedge}(\mathsf{S})))) \quad = \quad \tau(\mathsf{SABCBADC}) = \square \wedge \wedge \vee \wedge \wedge \vee \vee$$

Finite paperfolding words

$$
\begin{aligned}
\tau(h_\wedge(h_\wedge(h_\wedge(\mathtt{S})))) &= \tau(\mathtt{SABCBADC}) = \square\wedge\wedge\vee\wedge\wedge\vee\vee \\
\tau(h_\wedge(h_\wedge(h_\vee(\mathtt{S})))) &= \tau(\mathtt{SABCDADC}) = \square\wedge\wedge\vee\vee\wedge\vee\vee \\
\tau(h_\wedge(h_\vee(h_\wedge(\mathtt{S})))) &= \tau(\mathtt{SADCBABC}) = \square\wedge\vee\vee\wedge\wedge\wedge\vee \\
\tau(h_\wedge(h_\vee(h_\vee(\mathtt{S})))) &= \tau(\mathtt{SADCDABC}) = \square\wedge\vee\vee\vee\wedge\wedge\vee \\
\tau(h_\vee(h_\wedge(h_\wedge(\mathtt{S})))) &= \tau(\mathtt{SCBABCDA}) = \square\vee\wedge\wedge\wedge\vee\vee\wedge \\
\tau(h_\vee(h_\wedge(h_\vee(\mathtt{S})))) &= \tau(\mathtt{SCBADCDA}) = \square\vee\wedge\wedge\vee\vee\vee\wedge \\
\tau(h_\vee(h_\vee(h_\wedge(\mathtt{S})))) &= \tau(\mathtt{SCDABCBA}) = \square\vee\vee\wedge\wedge\vee\wedge\wedge \\
\tau(h_\vee(h_\vee(h_\vee(\mathtt{S})))) &= \tau(\mathtt{SCDADCBA}) = \square\vee\vee\wedge\vee\vee\wedge\wedge
\end{aligned}
$$

Infinite paperfolding words

Definition

Given an infinite sequence of instructions $I \in \{\wedge, \vee\}^{\omega}$, define

$$\mathbf{pf}(I) = \lim_{n \to \infty} \tau(h_{I_1}(h_{I_2}(\cdots h_{I_n}(\mathsf{S})\cdots)))$$

Infinite paperfolding words

Definition

Given an infinite sequence of instructions $I \in \{\wedge, \vee\}^\omega$, define

$$\mathbf{pf}(I) = \lim_{n \to \infty} \tau(h_{I_1}(h_{I_2}(\cdots h_{I_n}(\mathsf{S})\cdots)))$$

E.g., the regular paperfolding word is

$$\mathbf{pf} = \mathbf{pf}(\wedge\wedge\wedge\cdots).$$

Infinite paperfolding words

Definition

Given an infinite sequence of instructions $I \in \{\wedge, \vee\}^\omega$, define

$$\mathbf{pf}(I) = \lim_{n \to \infty} \tau(h_{I_1}(h_{I_2}(\cdots h_{I_n}(\mathsf{S})\cdots)))$$

E.g., the regular paperfolding word is

$$\mathbf{pf} = \mathbf{pf}(\wedge\wedge\wedge\cdots).$$

Fact

There is an uncountable family of paperfolding words.

$$h_{\wedge,0}(\mathsf{S}) = \mathsf{S}$$
$$h_{\wedge,0}(\mathsf{A}) = \mathsf{B}$$
$$h_{\wedge,0}(\mathsf{B}) = \mathsf{B}$$
$$h_{\wedge,0}(\mathsf{C}) = \mathsf{D}$$
$$h_{\wedge,0}(\mathsf{D}) = \mathsf{D}$$

$$h_{\vee,0}(\mathsf{S}) = \mathsf{S}$$
$$h_{\vee,0}(\mathsf{A}) = \mathsf{B}$$
$$h_{\vee,0}(\mathsf{B}) = \mathsf{B}$$
$$h_{\vee,0}(\mathsf{C}) = \mathsf{D}$$
$$h_{\vee,0}(\mathsf{D}) = \mathsf{D}$$

$$h_{\wedge,1}(\mathsf{S}) = \mathsf{SA}$$
$$h_{\wedge,1}(\mathsf{A}) = \mathsf{BC}$$
$$h_{\wedge,1}(\mathsf{B}) = \mathsf{BA}$$
$$h_{\wedge,1}(\mathsf{C}) = \mathsf{DC}$$
$$h_{\wedge,1}(\mathsf{D}) = \mathsf{DA}$$

$$h_{\vee,1}(\mathsf{S}) = \mathsf{SC}$$
$$h_{\vee,1}(\mathsf{A}) = \mathsf{BA}$$
$$h_{\vee,1}(\mathsf{B}) = \mathsf{BC}$$
$$h_{\vee,1}(\mathsf{C}) = \mathsf{DA}$$
$$h_{\vee,1}(\mathsf{D}) = \mathsf{DC}$$

We have four morphisms $\{h_{\wedge,0}, h_{\wedge,1}, h_{\vee,0}, h_{\vee,1}\}$ on alphabet

$$\Sigma = \{\mathsf{S}, \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{S}, \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\},$$

to construct prefixes of arbitrary paperfolding words.

We have four morphisms $\{h_{\wedge,0}, h_{\wedge,1}, h_{\vee,0}, h_{\vee,1}\}$ on alphabet

$$\Sigma = \{\mathsf{S}, \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{S}, \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\},$$

to construct prefixes of arbitrary paperfolding words.

Combine with the lemma.

Theorem

Given a DFAO T, there exists a DFAO M over $\{\wedge, \vee\} \times \{0, 1\}$ which computes

$$T(\mathbf{pf}(i_1 \cdots i_k)[0..n-1])$$

on input $i_1 \cdots i_k \in \{\wedge, \vee\}^*$ and $(n)_2$ for $0 \leqslant n < 2^k$.

All I want for Christmas is **an algorithm to detect a Sturmian word accepted by an ω-automaton**

Pierre BÉAUR,
joint works with Benjamin HELLOUIN de MENIBUS

LISN, Université Paris-Saclay

All I want for Christmas is
an algorithm to detect a Sturmian word
accepted by an ω-automaton

Pierre BÉAUR,
joint works with Benjamin HELLOUIN de MENIBUS

LISN, Université Paris-Saclay

What about an algorithm for paperfolding words?

ω-automata

Many kinds of ω-automaton:

- Büchi
- Rabin
- Streett
- Parity
- Muller

ω-automata

Many kinds of ω-automaton:

- Büchi
- Rabin
- Streett
- Parity
- Muller
  - Apply a deterministic automaton to the word.
  - Accept depending on the set of recurrent states.

ω-automata

Many kinds of ω-automaton:

- Büchi
- Rabin
- Streett
- Parity
- Muller
  - Apply a deterministic automaton to the word.
  - Accept depending on the set of recurrent states.

Sounds like transduction!

Fix a state $q \in Q$ of the Muller automaton (M).

q is recurrent (in $\mathbf{pf}(I)$)

Fix a state $q \in Q$ of the Muller automaton (M).

$$\begin{aligned} & q \text{ is recurrent (in } \mathbf{pf}(I)) \\ \iff \quad & \forall y, q \text{ occurs after pos. } y \end{aligned}$$

Fix a state $q \in Q$ of the Muller automaton (M).

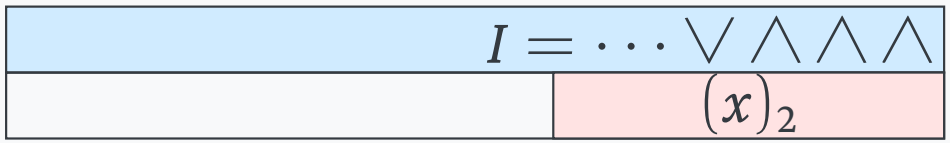
$$
\begin{aligned}
& q \text{ is recurrent (in } \mathbf{pf}(I)) \\
\iff \quad & \forall y,\, q \text{ occurs after pos. } y \\
\iff \quad & \forall y, \exists x,\, q \text{ occurs at pos. } x \geqslant y
\end{aligned}
$$

Fix a state $q \in Q$ of the Muller automaton (M).

$$
\begin{aligned}
& q \text{ is recurrent (in } \mathbf{pf}(I)) \\
\iff \quad & \forall y,\, q \text{ occurs after pos. } y \\
\iff \quad & \forall y, \exists x,\, q \text{ occurs at pos. } x \geqslant y
\end{aligned}
$$

Fact

There exists an DFA which accepts



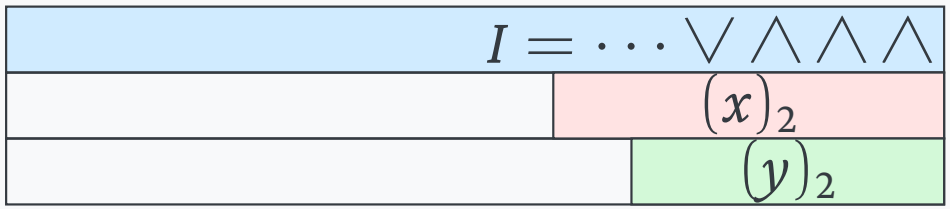
if M is in state q after reading $\mathbf{pf}(I)[0..x-1]$

Fix a state $q \in Q$ of the Muller automaton (M).

$$\begin{aligned} & q \text{ is recurrent (in } \mathbf{pf}(I)) \\ \iff \quad & \forall y,\, q \text{ occurs after pos. } y \\ \iff \quad & \forall y, \exists x,\, q \text{ occurs at pos. } x \geqslant y \end{aligned}$$

Fact

There exists an DFA which accepts



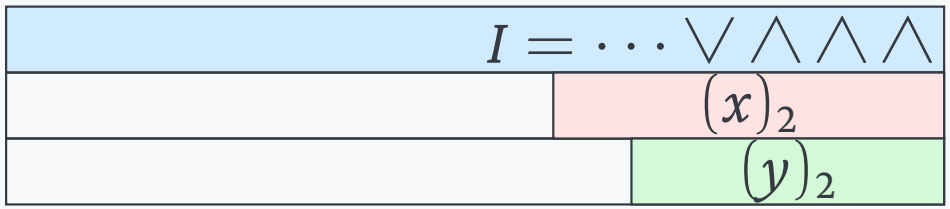
if M is in state q after reading $\mathbf{pf}(I)[0..x-1]$ and $x \geqslant y$.

Fix a state $q \in Q$ of the Muller automaton (M).

$$\begin{aligned} & q \text{ is recurrent (in } \mathbf{pf}(I)) \\ \iff\ & \forall y,\ q \text{ occurs after pos. } y \\ \iff\ & \forall y, \exists x,\ q \text{ occurs at pos. } x \geqslant y \end{aligned}$$

Fact

There exists an DFA which accepts



if M is in state q after reading $\mathbf{pf}(I)[0..x-1]$ and $x \geqslant y$.

Important! Convert DFA to ω-automaton before quantifying.

- We have constructed ω-regular languages

$$\mathsf{Recur}(q) := \{I \in \{\wedge, \vee\}^{\omega} \mid \text{state } q \text{ is recurrent on } \mathbf{pf}(I)\},$$

for all states q.

- We have constructed ω-regular languages

$$\mathsf{Recur}(q) := \{I \in \{\wedge, \vee\}^\omega \mid \text{state } q \text{ is recurrent on } \mathbf{pf}(I)\},$$

  for all states q.

- The set of recurrent states is $S \subseteq Q$ on $\mathbf{pf}(I)$ if I is in

$$\bigcap_{q \in S} \mathsf{Recur}(q) \cap \bigcap_{q \notin S} \overline{\mathsf{Recur}(q)}.$$

- We have constructed ω-regular languages

$$\mathsf{Recur}(q) := \{I \in \{\wedge, \vee\}^\omega \mid \text{state } q \text{ is recurrent on } \mathbf{pf}(I)\},$$

  for all states q.

- The set of recurrent states is $S \subseteq Q$ on $\mathbf{pf}(I)$ if I is in

$$\bigcap_{q \in S} \mathsf{Recur}(q) \cap \bigcap_{q \notin S} \overline{\mathsf{Recur}(q)}.$$

- We can construct an ω-automaton for

$$\{I \in \{\wedge, \vee\}^\omega \mid M \text{ accepts } \mathbf{pf}(I)\}.$$

Claim

There is an algorithm for detecting a paperfolding word in an ω-regular language.

Recap

- k-automatic sequences have k-regular partial sums
  - reduction mod $m \implies$ k-automatic,
  - finitely many values $\implies$ k-automatic,
  - linearly related to k-automatic $\implies$ k-synchronized
  - some partial sums are not synchronized
  - special case: Beatty sequences are synchronized

Recap

- k-automatic sequences have k-regular partial sums
  - reduction mod $m \implies$ k-automatic,
  - finitely many values $\implies$ k-automatic,
  - linearly related to k-automatic $\implies$ k-synchronized
  - some partial sums are not synchronized
  - special case: Beatty sequences are synchronized
- k-automatic sequences transduce to k-automatic sequences
  - three equivalent versions, first proved by Dekking
  - proof of monoid version

Recap

- k-automatic sequences have k-regular partial sums
  - reduction mod $m \implies k$-automatic,
  - finitely many values $\implies k$-automatic,
  - linearly related to k-automatic $\implies k$-synchronized
  - some partial sums are not synchronized
  - special case: Beatty sequences are synchronized
- k-automatic sequences transduce to k-automatic sequences
  - three equivalent versions, first proved by Dekking
  - proof of monoid version
- transduction proof generalizes beyond prefixes
  - transducing factors
  - paperfolding sequences
  - S-adic sequences?